PDE extension – simplified version

example models and comparison to Saldamli

Jan Šilar
jan.silar@lf1.cuni.cz

October 10, 2014

Simplified extension is introduced by example models first. It is than compared to Levon's work ([1], mostly chapter 4). Restrictions of this simplified extension are

- 1D models only
- only first derivative in both time and space

Simplified extension is subset of full proposed extension described in [2] (https://openmodelica.org/svn/OpenModelica/trunk/doc/PDEInModelica/doc/comparisonToSaldamli.pdf)

Support for this simple extension in OM should be implemented within my PhD.

1 Example models

1.1 Advection equation

```
model advection "advection equation"
  import PDEDomains.*
  parameter Real L = 1; // length
  parameter Real c = 1;
  parameter DomainLineSegment1D omega(l = L);
  field Real u(domain = omega, start = 0);
equation
  pder(u,time) + c*pder(u,x) = 0; //by default in omega.interior
  u = sin(2*3.14*time)              in omega.left;
end advection;
```

Listing 1: Advection equation in Modelica

- class `DomainLineSegment1D` from package `PDEDomains`

1.2 String equation

```
model string "model of a vibrating string"
  import PDEDomains.*;
  parameter Real L = 1; // length
  parameter Real c = 1; // tension/(linear density)
  parameter DomainLineSegment1D omega(l = L);
  field Real u( domain = omega, start = sin(-2*3.14/sqrt(c)*omega.x) );
  field Real v( domain = omega );
initial equation
  pder(u,time) = 0;
equation
  pder(u,time) - c*pder(v,x) = 0;
  pder(v,time) - pder(u,x) = 0;
  u = sin(2.0*3.14*time)              in omega.left;
  pder(u,x) = 0                       in omega.right;
end string;
```

Listing 2: String model in Modelica

- system of two equations avoiding second derivatives
- Dirichlet and Neumann BC

2 Comparison to Saldamli

2.1 Domain definition

Originally
see [1] – 4.3.1.1 and 4.3.1.2

Saldamli defines domain shape by listing its boundaries. Individual boundaries (points in 1D, curves in 2D resp. surfaces in 3D) are describes by shape-functions. Shape-function maps intervals ([0,1] for curves, [0,1]×[0,1] for surfaces) onto the boundary.

This approach is unnecessarily complicated in 1D. Other problems come in higher dimensions and are discussed in [2].

Alternative approach
We have several new built-in types.

Built-in `type Coordinate`:

```
type Coordinate = Real;
```

Built-in `type Domain` contains `type Region` to represent interior and boundaries of the domain. `Region` is nested in `Domain` to prevent instantiating `Region` outside a `Domain`.

```
type Domain
  type Region
    parameter Integer ndim; //dimension of the region
  end Region;
  parameter Integer ndimD; //dimension of the domain
  Coordinate coord[ndimD];
  replaceable Region interior; //main region of the domain
end Domain;
```

`type Domain` is extended by domain types for particular dimensions, e.g. in 1D:

```
type Domain1D
  extends Domain(ndimD = 1);
  type Region0D = Region(ndim = 0); //for boundaries
  type Region1D = Region(ndim = 1); //for interior
end Domain1D;
```

These types are extended by particular domains, e.g.:

```
type DomainLineSegment1D
  extends Domain1D;
  parameter Real l = 1; //length
  Coordinate x(name = "cartesian") = coord[1]; //alias for the coord
  Region1D interior(x in (0,l) /*or 0<x and x<l ??*/);
  Region0D left(x = 0); //boundaries
  Region0D right(x = l);
end DomainLineSegment1D;
```

New syntax (`x in (0,l)`) is used here to specify the range for the coordinate x within the region. I'm not sure about it, may be something better could be proposed.

Differential operators

4.3.2

Partial derivatives

Higher order derivatives are discussed here even though not supported for now.

Originally
see 4.3.2.1
e.g. $\frac{\partial u}{\partial t}$ .. `der(u)`, $\frac{\partial^2 u}{\partial x \partial y}$ .. `der(u,x,y)`

Problem
There is no way to write mixed time and space derivative $\frac{\partial^2 u}{\partial x \partial t}$ in this notation.

This notation doesn't agree with mathematics, where we have different operators for ordinary ($\frac{du}{dx}$) and partial ($\frac{\partial u}{\partial x}$) derivatives.

`der(u)` for partial time derivative is confusing. Ordinary derivative operator is used for total derivative in functions of several variables.

Alternative approach
$\frac{\partial u}{\partial t}$ .. `pder(u,time)`
$\frac{\partial^2 u}{\partial x \partial y}$ .. `pder(u,x,y)`

Now we can also write $\frac{\partial^2 u}{\partial x \partial t}$ as `pder(u,x,time)`

Accessing coordinates in der() operator

Originally
not discussed

Problem
Coordinates are defined within the domain type, but they are used in equations that are written outside domains. Thus they should be accessed using `domainName.` prefix (e.g. `omega.x`), which is tedious.

In the example in 4.3.2.2 in [1] the normal vector `n` is reached outside the domain class without `domainName.` prefix even thou it is defined in the domain. It is not explained how this is enabled.

Solution
Fields are differentiated with respect to coordinates only. Thus in place of second and following operands of `pder()` operator may be given only coordinates. So variables in this positions may be treated specially: coordinates of the domain of the field being differentiated may be accessed without the "`domainName.`" prefix here. If a variable of the same name exists in the model, they wont be in a conflict as the model variable can not appear in place of the second or following operand of `pder()`.

Perhaps usage of this shortened notation was intended even in the original extension but was not mentioned.

`In` operator

Just a remark: All equations containing a field variable (defined on a domain) hold on particular region of the domain. If the region is not specified (using "`in domain.region`") region `interior` is assumed implicitly.

Field literal constructor

originally
see 4.2.2, e.g.:

```
u = field(2*a+b for (a,b) in omega)
```
where iterator variables `(a,b)` exist only in constructor expression and represent coordinates in omega (probably `coord`, but may be `cartesian`, it is not clear from the document.)

problem

This syntactic feature is redundant (explained in next paragraph). It also has some problems in the full extension (see [2]). And the syntax suggested below is shorter and simpler anyway.

solution

Coordinate variables are also fields. (Coordinates vary their value over space as other fields.) Thus any expression depending on coordinates should be evaluated to another field. So we can write just

```
u = 2*omega.x+omega.y;
```

If coordinates are used often (not only in `pder()`), an alias for it may be defined to avoid "`omega.`" prefix, e.g.

```
coord Real x = omega.x;
```

But this would need new keyword `coord`.

References

[1] Levon Saldamli. *A High-Level Language for Modeling with Partial Differential Equations.* PhD thesis, Department of Computer and Information Science, Linköping University, 2006.

[2] Jan Šilar. Pde extension – changes over levon's extension. https://openmodelica.org/svn/OpenModelica/trunk/doc/PDEInModelica/doc/comparisonToSaldamli.pdf.